ASGriDS: Asynchronous Smart-Grids Distributed Simulator

Takai-Eddine Kennouche[1], Florent Cadoux[2], Nicolas Gast[1], and Benoît Vinot[3]

[1]Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP*, LIG, 38000 Grenoble, France
{takai-eddine.kennouche, nicolas.gast}@univ-grenoble-alpes.fr
[2]Univ. Grenoble Alpes, CNRS, Grenoble INP*, G2ELab, 38000 Grenoble, France
florent.cadoux@grenoble-inp.fr
[3]Roseau Technologies, 38240 Meylan, France
benoit.vinot@roseautechnologies.com

***Abstract*—We present ASGriDS, an asynchronous Smart Grid simulation framework. ASGriDS is multi-domain, it simultaneously models the power network along with its physical loads/generators, controllers, and communication infrastructure. ASGriDS provides a unified workflow in a pythonic environment, to describe, run and control complex SmartGrid deployment scenarios. ASGriDS is an event-driven simulator that can run in either real-time or accelerated real-time. As it is modular and its components interact asynchronously, it can run either locally on a distributed infrastructure, also in hardware-in-the-loop setups, and on top of emulated/physical communication links. In this paper, we present the design of our simulator and we demonstrate its use with a generation control problem on a low voltage network. We use ASGriDS to deploy a real-time controller based on optimal power flow, on top of TCP and UDP based communication network, under various packet loss conditions.**

***Index Terms*—real-time simulation, asynchronous, co-simulation, distributed simulation, HIL, Smart Grid Communication**

I. Introduction

The Smart Grid is a cyber - physical energy system, where the physical processes of a power system are strongly integrated with a communication network infrastructure and a set of possibly complex controllers. This vision of the modern power grid is motivated by the new challenges that are currently raised by the need to massively integrate distributed renewable energy generators. In addition to that, the Smart Grid is expected to satisfy growing reliability and efficiency demands, due to the fast transition to electrical power in public transportation systems and the growing usage of electrical vehicles.

The Smart Grid needs to solve a variety of reliability and efficiency challenges [1]. In this context, Information and Communication Technologies (ICT) come into play as an enabler for more intelligence to manage a complex set of operational, monitoring and optimization constraints, and allow for innovative solutions and novel applications. On the other hand, understanding the ICT infrastructure's performance limits, such as managing packet transmission delays, is crucial for safety and stability of the power grid and efficiency of control, monitoring and recovery of the network.

New techniques are needed to properly prototype, test and validate a modern Smart Grid. Traditional testing and domain-specific simulation techniques and software tools only allow for partial representation of the system; for instance, there exists many domain-specific tools for the simulation of either the physics of a power grid, or for the simulation of telecommunication networks, but none of these tools are capable of embracing the complexity of a Smart Grid. Such tools are simply not sufficient to trace the complex behavior of intertwined power and telecommunication networks: for this purpose, multi-domain frameworks must be developed.

ASGriDS is developed to provide a modular and generic simulation description environment, that relies on an event-driven architecture to model power network components, and an asynchronous communication module that is scalable and reliable. Our framework is capable of running both local and distributed simulations. The nodes are asynchronous in the sense that they don't rely on heavy explicit synchronization, but they rely instead on the fact that their execution is driven by local clocks that are globally synchronous enough. This choice of implementation allows for realistic modeling of a real asynchronous Smart Grid behavior. It is also technically required to be able to account for real-time execution and Hardware-In-the-Loop (HIL) integration. In addition, our framework is capable of modeling complex communication network behavior, through the use of Linux network emulation that permits rich tuning capabilities of delay, jitter, loss, duplication, reordering, corruption and rate. This allows the modeling of wired and wireless networks [2].

The rest of the paper is organized as follows. In Section II, we review the state of the art of Smart Grid simulators, and put that in contrast with our own proposal. We describe the architecture and design choices of ASGriDS in Section III and evaluate its performance in Section IV. Finally, we present in Section V a case study concerning photovoltaic production control, in a low voltage network and lossy communication.

II. Related Work

The Electric Power and Communication Synchronizing Simulator (EPOCH) [3], is a simulation framework that aims to study complex scenarios involving combined power and communication networks. EPOCH runs in a federated simulation environment, using PSCAD and PSLF for power simulations and UC Berkeley's Network Simulator 2 (NS-2) [4] for communication network simulations, with fixed time-stepping clock synchronization.

The Global Event-Driven Co-Simulation Framework (GECO) [5] is another framework that targets power system monitoring and control using a communication network. It also combines PSLF power system simulator and NS-2, but differs than EPOCH in that it uses a global event-driven co-simulation environment. GECO ensures full synchronization of events, by encapsulating dynamic power system simulation runs in discrete events queued in a global events queue and ordered with the network simulator events.

Another co-simulation framework is the Integrated co-Simulation of Power and ICT systems for Real-time Evaluation (INSPIRE) [6]. INSPIRE co-simulation environment consists of running DIgSILENT PowerFactory for power simulation, in conjunction with OPNET Modeler with a model of IEC 61850 communication protocol. Both are orchestrated with dynamic time-stepped clock.

A different simulation approach is followed in MECSYCO [7], where the authors adhere to strict formalism using a Discrete Event System Specification (DEVS) to describe their multi-agent system. The interaction of their systems, though, is always strictly synchronized and does not provide any real time capabilities or the possibility of interaction with components outside the simulator.

T-RECS [8] is another interesting simulator, its stated objectives are close to what we intend to develop in ASGriDS, in that it incorporates network emulation for more realistic control over the communication medium, and provides an API for the interaction with pre-existing components. However, as pointed by their authors in [8], it cannot scale to more than ten or a few tens of nodes, whereas ASGriDS can scale to hundreds.

It is noticeable that a limiting factor for realism and scalability of simulation, is the constraints of strong event/time synchronization among simulation components, and/or the implementation of a global event scheduler. Time synchronization can be a very fit choice, if the goal is to model synchronous systems, and/or to guarantee reproducible simulations and easy testing/debugging of certain system's aspects, but it falls short to capture the nature of an actually asynchronous system: such a system may only be accurately modelled by means of an asynchronous simulation framework, without a compromise in terms of accuracy [9]. On the other hand, a global event scheduler, does not allow for any real-time execution, and limit the possibility to interact with other real-time components of the system.

Our work is intended to provide scalability of simulation, realism of modeling and flexibility of scenario description, through an asynchronous design and event-driven node model.

- **Asynchronism**: ASGriDS does not enforce explicit time synchronization of the different nodes. Each node is driven by its local system's clock. Asynchronous communication is also used to decouple network I/O operations from nodes internal operations.
- **Real-time**: The simulation nodes all run in real-time, in the sense that they are driven by system's real-time clock. This allows the nodes to interact with the outside world — for example, with a real communication infrastructure.
- **Scalability**: Scalability comes as a consequence of the modularity, event-driven and asynchronism of our architecture, as we show in section IV. The socket-based communication module, also permits scalability both locally and on a distributed deployment, without sacrificing of its key features of asynchronism and real-time execution.

Our framework provides the necessary architecture for complex modeling of power/communication interactions, using either emulation of the ICT network, simulation models or interaction with real setup.

III. Design of ASGriDS

The general overview of ASGriDS's architecture is outlined in Figure 1. The components shown in the figure represent the different aspects of a Smart Grid, whether it is the power networks simulation, the optimization and control or the network communication. These components, as seen in the figure, interact with each other at two separate but complimentary planes, a **Deployment Plane** and a **Communication Plane**. In this section, we describe these two operational planes of the simulator, after which we discuss the various building blocks of ASGriDS and how it integrates both network emulation for ICT modeling, and electrical network simulation.

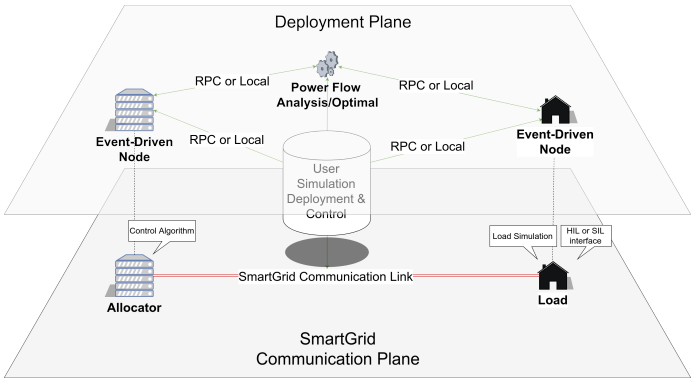

Fig. 1. Simulation Deployment Architecture

A. Deployment Plane

At the Deployment Plane level, ASGriDS deploys and orchestrates the simulation, the traffic flowing between the components at this level is exclusively the traffic necessary to control the various distributed elements of the simulator.

The distinction between this and the actual simulation traffic is important, for instance in a distributed scenario one might want to change the behavior of a controller reployed remotely, or provide a remote node with a new production profile, or schedule some events, this will incur network traffic generated by the object-proxying mechanism that we rely on, this traffic isn't what we would like to observe or interact with, but it is necessary for simulation deployment. On a local scenario, this traffic will simply be the communication between system processes/threads. ASGriDS basically triggers a "remote object" mode when run distributively that allows the user to handle simulation components, and network nodes as if they were running locally. This was possible using the concept of object-proxying and remote method invocation, and their implementation provided by the free and open-source library `RPyC` [10].

B. Smart Grid Communication Plane

At the Smart Grid Communication Plane level, ASGriDS simulates the actual communication network of interest for the end-users, where the traffic that is flowing is supposed to represent a real Smart Grid deployment traffic. As we mentioned before, ASGriDS can be deployed on a variety of ICT models, whether emulated, simulated or real. By default, it allows very easy and flexible manipulation of the communication link between simulated nodes, through binding to a linux local network interface (local loop or virtual interface). Network impairments (delay, packet loss, duplication…) can then be controlled by the user through an interface to linux traffic shaping (*tc*) [11] utility and the *netem* module [2].

C. Event-Driven Real-Time Node

The basic component of the proposed framework is a real-time Event-Driven "Node", that is shown in Figure 2. The "Node" component of ASGriDS, is an event-driven. It can be deployed as a process, a thread or a remote object in a distributed simulation. It interacts with its environment is callback-based, for instance when handling network events as in Figure 2. Most importantly, a node can be easily deployed to interact with any real-time source of events, such as hardware component (*e.g.* real communication network) or software implementation of a certain behavior, that can already be available to the user through external libraries, either to test/validate software/hardware components or to implement new ones in a configuration as close to reality as possible. This renders the framework fit for a variety of HIL and Software-In-the-Loop (SIL) setups.

D. Asynchronous Communication

In order for ASGriDS's "Node" to truly communicate in a network, it is supported by a communication layer module, and the interaction between the two happens through a callback mechanism, as shown in Figure 2. In designing the module, our objective is a complete decoupling from the node through asynchronism and non-blocking network Input/Output operations, and pluggability in the sense that the communication layer can be modified/extended and easily plugged into a "Node".

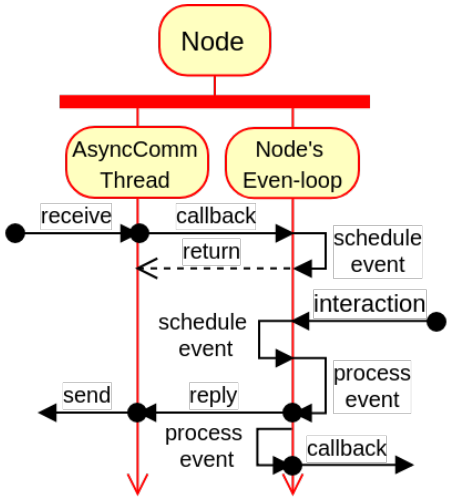

Fig. 2. Node — Network interaction

We chose to implement this asynchronous communication ("AsyncComm") module around `asyncio` [12]. `asyncio` is a standard python library, it features flexible and industry-proven asynchronous API, and provides the necessary blocks to implement a communication layer that is completely asynchronous and modular, through its concurrent execution model and customizable event-loops.

E. Electrical Simulation Integration

In ASGriDS, electrical simulation is handled as a background service, controlled at the Deployment Plane, and interfacing with various other simulation components at the Smart Grid Communication Plane. For instance, this service keeps track of the electrical network state by solving load flow equations in real time. It provides an interface to other nodes so that they can access their local electrical state (*i.e.* voltage).

In our experiments (that we present in Section IV and V), we implement this component around `pandapower`, a free and open-source python library that provides access to various power system simulations such as load flow and optimal power flow solvers.

IV. Performance of ASGriDS

In this section, we study the scalability of ASGriDS in terms of CPU and memory consumption. Our assumption is that for low-voltage distribution networks and micro-grids, a scalable simulator should be able to handle in the order of hundreds of network nodes. For an entire distribution network, one to two additional orders of magnitude may be necessary.

To measure the scalability of ASGriDS, we design a set of experiments according to the architecture in Figure 1. Every component of the simulation is run as an independent computation node, that is either part of the simulated grid (Smart Grid Communication Plane), *i.e.* a load or a generator (PV or

otherwise) or part of the simulation management (Deployment Plane). At the simulation plane, each node simulates either a network load or a PV generator. These nodes communicate with a special node acting as a central allocator as follows: each node reports its voltage measurements and current production/consumption levels, and receives production setpoints in the case if PV generators. The allocator is running a control algorithm, that observes the network state through the voltage measures and acts accordingly to keep the network stable within a defined voltage limit.

A. System Resources Consumption

We define a set of benchmarks by using `pandapower`'s implementation of Washington Case300 network as a base-case. By default this network includes 193 loads (which corresponds to 193 nodes in our simulator). To vary the number of nodes of the network, we add or remove randomly nodes of this network. Every simulated node will be generating fake producation/consumption values in the electrical network, these values will be reported to an allocator (controller) along with voltage measurements gathered from `pandapower`'s power flow analysis, the allocator will then `pandapower`'s optimal power flow solver to generate new setpoints for the nodes.

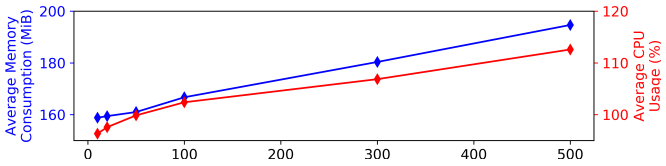

Fig. 3. ASGriDS's CPU and Memory consumption for various network sizes

In Figure 3, we report the maximal memory usage of our simulator during the first 3 minutes of simulation as a function of of the number of nodes deployed. We observe that the memory consumption remains limited and roughly equals 140 MB plus 50 kB per simulated node. Note that we also run the tests with other scenarios than the Washington Case300 and obtain similar values.

In all of our experiments, the CPU usage for our simulator is mainly dominated by the load flow. To verify that `pandapower` was fit for the need of our simulator, we measured the time taken by this library to solve load flow and optimal power flow equations. Our measures show that solving one load flow takes less than 100 ms while computing an optimal power flow takes a delay between 0.5 s for small networks, and 2.5 s for ∼200 nodes network.

V. Case Study: OPF-based Controller on a Lossy Network

To demonstrate the capabilities of ASGriDS, we present a case study where we deploy CIGRE's low voltage network [13] in ASGriDS. CIGRE's loads become "Nodes" capable of PV power generation, by incorporating load and production profiles from [14]. The low voltage network consists of 37 nodes distributed over residential, industrial and commercial sub-networks, and includes 6 residential loads, 8 commercial and one industrial. This benchmark encapsulates the most relevant technical aspects of a real electrical grid, and it is developed for the purpose of modeling and simulating modern micro-grids efficiently [15].

In the deployed simulations, every load from the CIGRE network corresponds to a "Node" in ASGriDS, running 24h recorded consumption and PV production profiles from [14], read from a database. In the simulations, for practical reasons, we simulate the systems in accelerated real time (x300) so that 24h simulated time corresponds to 5min realtime.

An allocator (controller) "node" runs the control loop as described in the petri net [16] diagram in Figure 4 and submits new PV production setpoints to all concerned nodes every cycle (15min simulation, 3s accelerated realtime). A power flow solver runs in the background (using `pandapower`) to provide timely voltage measurements to nodes (Deployment Plane). Each node reports this voltage value along with its consumption/production, to the allocator (Communication Plane). When a node receives a new setpoint, it implements the maximum between the new setpoint and the current maximum production capacity.

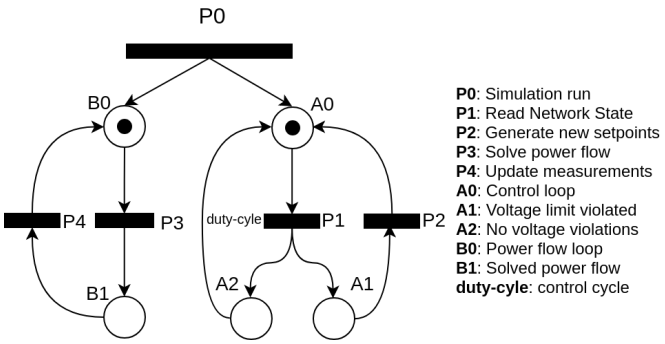

Fig. 4. Power flow analysis and production control, running in parallel

We use the experiments to observe the performance of an Optimal Power Flow based controller described in section V-A. The controllers will be operating with a communication packet loss ranging between no loss (ideal communication), and 60 % of packet loss (extreme conditions). The network is deployed on an emulated link, and losses are controlled through linux's *netem* module. Recall that the network losses affect the transmission between the allocator node and the loads. The communication between the nodes and the power flow analyzer is not affected by the losses as such communications are done through the deployment plane.

A. Optimal Power Flow Controller

We implement the Optimal Power Flow (OPF) control loop around the OPF formulation in Equation (1). Where P is the

production capacity every PV producer.

$$\begin{aligned} \max \quad & \sum_{i \in \text{PV generators}} P_i \\ \text{subject to } & V_{\min} \leqslant V_{g,i} \leqslant V_{\max}, j \in \text{bus} \\ & L_k \leqslant L_{\max,k}, k \in \text{transformer} \\ & L_l \leqslant L_{\max,l}, l \in \text{line} \\ & P_i \leqslant P_{i,\max} \end{aligned} \tag{1}$$

In our experiments, we use `pandapower` to solve the OPF (with default parameters). The constraints are 100 % loading for branch elements, and $\pm 5\%$ p.u. around nominal voltage for bus voltages. The value of $P_{i,\max}$ is a forecast of what we think could be the maximal production of the PV panel i.

B. Numerical Results

In Figure 5 we plot the performance of the OPF controller, in terms of the rate of voltage violations (voltage outside 5% of nominal value) detected from all measurements in the network, as a function of various losses configure in the communication channel, while using either TCP or UDP as a transport protocol. The baseline (in red), is when there is no control. We observe that the OPF in TCP mode consistently outperforms the OPF in UDP mode, although the difference is small for packet loss of 0%, 10%, 20% and 30% the performance is very good in both cases. With 60% packet loss, we notice that the controller manages to perform better with UDP, we explain that by the fact that TCP might exhibit high retransmission rate with such high packet loss, thus delivers outdated measurements and control information, the effect that is absent in UDP.

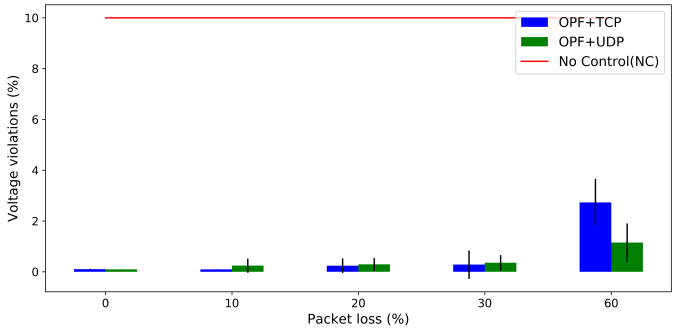

Fig. 5. Voltage violations with TCP/UDP communication on a lossy network

VI. Conclusion

We have presented in this work, ASGriDS, a framework for the design, prototyping and testing of Smart Grid deployment scenarios through distributed and scalable simulation. In ASGrIDS, real-time control strategies can be implemented, and deployed on a communication network. ASGriDS allows the integration of emulated communication links, and/or the deployment over a physical ICT infrastructure. It provides a consistent workflow for describing complex multi-domain Smart Grid deployment scenarios. It is scalable and flexible, through a modular and asynchronous design. ASGriDS permits the various simulated components to exhibit behavior at various level of accuracy and complexity, and allows the integration of simulated power-network components, and the deployment of control strategies on physical communication hardware, or in combination with other hardware and software network components. A typical use-case is demonstrated, analyzing control performance over a lossy link, with two modes of communication, TCP and UDP. ASGriDS is publicly available on a github repository [17] together with the results described in this paper. We plan to continue its development, especially in distributed setups, with real ICT infrastructure and in HIL configurations. In this paper, we illustrated the capabilities of ASGriDS with a simple OPF controller described in (1). For future work, we plan to use ASGriDS to develop controllers that are fault tolerant and that can deal with measurement uncertainty and non-ideal communication.

References

[1] K. Moslehi and R. Kumar, "A Reliability Perspective of the Smart Grid," *IEEE Transactions on Smart Grid*, vol. 1, no. 1, pp. 57–64, Jun. 2010.

[2] S. Hemminger *et al.*, "Network emulation with NetEm," in *Linux Conf Au*, 2005, pp. 18–23.

[3] K. Hopkinson, X. Wang, R. Giovanini, J. Thorp, K. Birman, and D. Coury, "EPOCHS: A platform for agent-based electric power and communication simulation built from commercial off-the-shelf components," *IEEE Transactions on Power Systems*, vol. 21, no. 2, pp. 548–558, May 2006.

[4] "The Network Simulator - ns-2," https://www.isi.edu/nsnam/ns/.

[5] H. Lin, S. Member, S. Veda, E. Shukla, L. Mili, S. Member, J. Thorp, and L. Fellow, *GECO: Global Event-Driven Co-Simulation Framework for Interconnected Power System and Communication Network*.

[6] H. Georg, S. C. Müller, N. Dorsch, C. Rehtanz, and C. Wietfeld, "INSPIRE: Integrated co-simulation of power and ICT systems for real-time evaluation," in *2013 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct. 2013, pp. 576–581.

[7] J. Vaubourg, Y. Presse, B. Camus, C. Bourjot, L. Ciarletta, V. Chevrier, J.-P. Tavella, and H. Morais, "Multi-agent Multi-Model Simulation of Smart Grids in the MS4SG Project," in *Advances in Practical Applications of Agents, Multi-Agent Systems, and Sustainability: The PAAMS Collection*, ser. Lecture Notes in Computer Science, Y. Demazeau, K. S. Decker, J. Bajo Pérez, and F. de la Prieta, Eds. Springer International Publishing, 2015, pp. 240–251.

[8] J. P. Achara, M. M. Maaz, W. Saab, R. Rudnik, J.-Y. Le Boudec, and L. E. Reyes Chamorro, *T-RECS: A Virtual Commissioning Tool for Software-Based Control of Electric Grids – Design, Validation, and Operation*. ACM, 2018.

[9] S. Ghosh, "The Role of Modeling and Asynchronous Distributed Simulation in Analyzing Complex Systems of the Future," *Information Systems Frontiers*, vol. 4, no. 2, pp. 161–177, Jul. 2002.

[10] "RPyC - Transparent, Symmetric Distributed Computing — RPyC," https://rpyc.readthedocs.io/en/latest/index.html.

[11] B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. Schroeder, J. Spaans, and P. Larroy, "Linux advanced routing & traffic control," in *Ottawa Linux Symposium*, vol. 213, 2002.

[12] "Asyncio — Asynchronous I/O — Python 3.7.3 documentation," https://docs.python.org/3/library/asyncio.html.

[13] "Benchmark Systems for Network Integration of Renewable and Distributed Energy Resources, version 7, CIGRE Task force C6.04.02," 2011.

[14] A. N. Espinosa and L. Ochoa, "Dissemination document low voltage networks models and low carbon technology profiles," *The University of Manchester*, 2015.

[15] S. Papathanassiou, N. Hatziargyriou, and K. Strunz, "A BENCHMARK LOW VOLTAGE MICROGRID NETWORK," p. 9.

[16] J. L. Peterson, "Petri Nets," *ACM Computing Surveys*, vol. 9, no. 3, pp. 223–252, Sep. 1977.

[17] "Takienn/asgrids," https://github.com/takienn/asgrids/tree/appeec19.